Análise do Algoritmo de Cardume de Peixes em Problemas Dinâmicos com Domínio Contínuo

Rafael Henrique Vincence

Joinville

2016

Rafael Henrique Vincence

Análise do Algoritmo de Cardume de Peixes em Problemas Dinâmicos com Domínio Contínuo

Relatório de Trabalho de Conclusão de Curso (TCC) apresentado ao Curso de Graduação em Ciência da Computação, da Universidade do Estado de Santa Catarina (UDESC), como requisito parcial da disciplina de Trabalho de Conclusão de Curso.

Orientador: Profº Rafael Stubs Parpinelli

Joinville

2016

Rafael Henrique Vincence

Análise do Algoritmo de Cardume de Peixes em Problemas Dinâmicos com Domínio Contínuo

Relatório de Trabalho de Conclusão de Curso (TCC) apresentado ao Curso de Ciência da Computação da UDESC, como requisito parcial para a obtenção do grau de BACHAREL em Ciência da Computação.

Aprovado em 2 de Junho de 2016

BANCA EXAMINADORA

Prof^{o} Rafael Stubs Parpinelli

Prof^{o} Guilherme Koslovski

Prof^{o} Chidambaram Chidambaram

Resumo

A maioria dos desafios encontrados na vida real são problemas complexos com uma grande gama de variáveis e por vezes dinâmico. Sendo assim, cada vez mais tem-se a necessidade da criação de meta-heurísticas para solucionar problemas com estas características. A Natureza, por sua vez, resolve instintivamente esses tipos de problemas e por esse motivo algoritmos bioinspirados tem sindo amplamente utilizados na resolução de problemas dinâmicos com domínio contínuo. Os aspectos e interações de uma colônia de animais são pontos relevantes na otimização desses problemas, como o comportamento individual e comunitário, que contribuem em problemas de larga escala e não estacionários. Neste trabalho o foco é a análise e aplicação do algoritmo de cardume de peixes (*Fish School Search*, FSS) em *benchmarks* dinâmicos de domínio contínuo.

Palavras-chave: *Inteligência de enxames*, *Algoritmos bioinspirados*, *Problemas dinâmicos com domínio contínuo*, *Fish School Search*.

Conteúdo

Lista de Figuras

Lista de Quadros

Lista de Abreviaturas

BA	*Artificial bee Colony*
BFO	*Computação Natural*
CN	Computação Natural
DE	*Diferential Evolution*
EA	*Evolutionary Algorithms*
FA	*Computação Natural*
FSS	*Fishing School Shearch*
IS	*Inteligence Swarm*
MP	*Moving Peaks*
OA	Operador de Alimentação
OMCI	Operador de Movimentação Coletiva Instintiva
OMI	Operador de Movimentação Individual
OMVC	Operador de Movimentação Volátil Coletiva
OP	*Ocillating Peaks*
PSO	*Parocle Swarm Optimization*

1 Introdução

Na sociedade existem vários tipos de problemas que possuem uma grande dificuldade de obterem resultados satisfatórias em um tempo factível. Esses problemas tendem a ter uma grande dimensionalidade e estarem sempre se alterando, o que os tornas problemas complexos de serem otimizados (OLIVEIRA; SILVA; ALOISE, 2004). Na Natureza pode-se observar também uma constante mudança no ambiente, de forma que seus integrantes tenham que se adaptar a essas mudanças para poderem sobreviver. Sendo assim, na Natureza existe uma grande fonte de inspiração para o desenvolvimento de novas tecnologias para solucionar essa classe de problemas, particularmente na Ciência da Computação, dentro da área de Computação Natural, que é responsável pela criação de diversos algoritmos (CASTRO, 2007). A partir da observação e compreensão do comportamento de animais ou colônia de animais (fenômenos naturais) foram desenvolvidas várias tecnologias em diferentes aplicações (ROZENBERG; BCK; KOK, 2011), pois a natureza é capaz de trabalhar com problemas de alta complexidade e grande dimensionalidade, motivando o desenvolvimento de algoritmos bioinspirados na área de otimização (ANDRÉ; STUBS PARPINELLI, 2015).

A característica que vem chamando atenção para a área de otimização na computação é justamente o fato dos problemas estarem em contante alteração e uma solução ótima, que foi encontrada em um instante de tempo anterior, pode não ser mais suficientemente boa para o mesmo problema no futuro. Isso é uma questão muito importante, pois a maioria dos algoritmos usados na otimização de problemas leva muito tempo para encontrar uma solução factível, por uma busca exaustiva e, se em pouco tempo ela perder a validade, o sistema de otimização não pode ser aplicado em ambientes dinâmicos (MORRISON, 2003). Por esse motivo, para otimizar esses problemas são utilizados os algoritmos bioinspirados, que por sua vez possuem vários operadores e características relevantes que podem ser aplicados e estudados em separado e assim estipular a relevância de cada um.

Na literatura são encontrados diversos exemplos de algoritmos com inspiração natural, que podem se basear no comportamento de uma colônia de animais por busca de alimento, na interação que eles podem realizar entre si em sua colônia, ou até na própria evolução das espécies. Uma das primeiras meta-heurísticas inspiradas na natureza, em especial na Biologia, são os Algoritmos Genéticos (*Genetic Algorithms*) (HOLLAND, 1975). Esta abordagem se ba-

seia na teoria da evolução proposta por Darwin em que os indivíduos mais bem adaptados tem maiores chances de sobreviver e passar seu material genético adiante, essa classe de algoritmos é chamada de computação evolucionária (*Evolutionary Algorithms* - EA) que possuem rotinas específicas. Outro algoritmo que entra nessa classe é o de Evolução Diferencial (*Diferential Evolution* - DE), que também possui uma rotina de seleção, cruzamento e mutação.

Dentre outros algoritmos bioinspirados existe uma classe que se baseia no comportamento simples de cada indivíduo separadamente e que em conjunto pode-se gerar um comportamento mais complexo, ou seja, um comportamento emergente. E essa classe é chamada de algoritmos de Inteligência de Enxame (*Inteligence Swarm* - IS) (PARPINELLI; LOPES, 2011). Entre eles pode-se citar algoritmos que tem sua aplicação a problemas contínuos, como o algoritmo baseado no comportamento de uma colônia de bactérias na busca por alimentos, o *Biomimicry of bacterial foraging* (BFO) (PASSINO, 2002), o algoritmo baseado no comportamento das colônias de morcegos e sua eco-localização para atacar um presa e desviar de alvos durante o voo, o *Bat Algorthm* (YANG; HOSSEIN GANDOMI, 2012). O algoritmo inspirado no comportamento coordenado do movimento de cardumes de peixes e revoada de pássaros, o Algoritmo de Otimização por Enxame de Partículas (*Partical Swarm Optimization*, PSO) (EBERHART; KENNEDY et al., 1995), dentre outros.

Um algoritmo que será o foco do trabalho, é o algoritmo baseado no comportamento de um cardume de peixes, o *Fish School Search Optimization* (FSS) (CARMELO FILHO, 2008), pois possuí operadores evolutivos eficientes na manutenção da diversidade quando é percebido uma piora na evolução e, na intensificação caso contrário. Suas principais características estão nos seus operadores evolutivos e na influência que cada um deles tem no processo de otimização (C JA FILHO, 2009), sendo eles: Operador de movimento individual; Operador de alimentação; Operador de movimento coletivo instintivo; e o operador de movimento volátil coletivo. O operador de movimento volátil coletivo tem uma influência maior na resolução de problemas dinâmicos e contínuos, pelo fato de expandir e contrair a busca do cardume de peixes, dependendo do nível de melhoramento das soluções em relação a experiências recentes, o que ajuda na manutenção da resiliência da busca. Outro modelo desse algoritmo possui outros operadores evolutivos (MADEIRO, 2011), que são: Operador de memória; e o Operador de divisão da escola de peixes. O operador de movimentação volátil foi aplicado no PSO para melhorar o desempenho em problemas dinâmicos, sendo criado uma nova versão, o *Volitive* PSO (CAVALCANTI-JÚNIOR, 2011), que obteve bons resultados aplicado a essa classe de problemas.

A literatura apresenta uma vasta quantidade de trabalhos que estudam a aplicação

desses algoritmos bioinspirados em problemas dinâmicos com domínio contínuo. Entre eles pode-se citar o AG que foi analisa em seus componentes em separado, não somente na sua performance (RAND; RIOLO, 2005), o ...

Como não se pode ter uma solução ótima a todo momento em problemas dinâmicos, o tempo de execução é considerado como uma unidade discreta. Geralmente um dos fatores limitantes em uma aplicação acaba sendo o tempo de execução sendo necessário manter um equilíbrio entre a qualidade da solução e o tempo de execução do algoritmo (LI; YANG; KANG, 2006). Apesar dos algoritmos bioinspirados serem capazes de encontrar boas soluções para problemas reais, eles tendem a perder a eficiência quando aplicados a problemas de larga escala. Esta característica indesejável é conhecida por "maldição da dimensionalidade" (*curse of dimensionality*) (BELLMAN; DREYFUS, 2015). Isso ocorre devido ao crescimento exponencial do espaço de busca de acordo com as dimensões do problema.

O grande número de dimensões aumenta a dificuldade dos algoritmos em manter soluções aceitáveis. A qualidade da otimização de um algoritmo depende do equilíbrio dos componentes de diversificação e intensificação (BOUSSAÏD; LEPAGNOT; SIARRY, 2013), pois a diversificação é responsável pela exploração do espaço de busca como um todo e a intensificação é responsável pela acurácia da resposta. A natureza evoluiu para manter o equilíbrio de diversidade e na otimização é possível usar alguns componentes de controle para sua preservação. A literatura aponta diversas estratégias para este controle de diversificação, sendo algumas: *fitness sharing, clearing, crowding, deterministic crowding, probabilistic crowding* e *restricted* (ANDRÉ; STUBS PARPINELLI, 2015).

Na literatura, os algoritmos bioinspirados são aplicados a diversos tipos de problemas dinâmicos, sendo eles contínuos ou discretos, então para poder avaliar os algoritmos, utiliza-sa diversas funções *benchmarks* (MOSER, 2007), como por exemplo a função de Movimentação de Picos (*Moving Peaks* - MP). A qualidade da solução sendo otimizada pode ser mensurada pela sua aptidão, ou seja, quão interessante é o valor encontrado e quão bem o algoritmo se adapta a uma mudança do ambiente. Assim, uma solução com uma boa aptidão (*fitness*) durante o processo de evolução é considerada válida e é chamada de solução factível. Durante a otimização dos *benchmarks* podem aparecer diferentes tipos de dinamismo, como por exemplo: na função objetivo; no domínio das variáveis; no número de variáveis; nas restrições; ou outros parâmetros.

Na realização desse trabalho foram estipulados objetivos para determinar os pas-

sos a serem seguidos, na próxima seção serão apresentados esses objetivos para estruturar os capítulos do trabalho.

1.1 Objetivo

Este trabalho tem como objetivo analisar da eficiência do FSS em problemas dinâmicos com domínio contínuos e com alta dimensionalidade. A proposta central é analisar cada um dos operados evolutivos dos algoritmos e determinar a relevância de cada um para a otimização dessa classe de problemas. Para isso será feito uma análise comparativa dos algoritmos que tem relevância na otimização desses problemas, indicando os pontos positivos e negativos de cada um. Para atingir o objetivo principal alguns objetivos específicos foram traçados:

- Revisão bibliográfica dos conceitos da computação natural e meta-heurísticas bioinspiradas;
- Análise aprofundada do FSS e seus operadores;
- Revisão bibliográfica dos operadores evolutivos existentes;
- Estudo dos algoritmos evolutivos de inteligência de enxame para verificar a relevância dos operadores existentes e realizar uma análise comparativa
- Levantamento e descrição de funções dinâmicos para serem aplicados nos experimentos;
- Desenvolver um algoritmo utilizando os operadores evolutivos alternadamente;
- Realizar experimentos computacionais com o algoritmo e os problemas selecionados;
- Coleta e análise dos resultados dos experimentos.

1.2 Estrutura do Trabalho

O trabalho está organizado em 6 Capítulos, incluindo a introdução

No segundo capítulo é apresentada uma definição dos problemas que serão estudados neste trabalho, esquematizando suas características principais, suas dificuldades e um método de avaliação dos algoritmos que será utilizado para realizar a otimização.

No terceiro capítulo é feita uma revisão do estado da arte dos algoritmos bioinspirados e as suas características principais, sendo separados em dois grupos: os algoritmos evolutivos e os de inteligência de enxames. Por fim, é apresentado uma revisão das técnicas de intensificação e diversificação encontradas na literatura.

No quarto capítulo é apresentada uma revisão dos trabalhos encontrados nessa área de pesquisa, onde pode ser analisado outras aplicações em problemas dinâmicos e com isso definir quais os pontos mais relevantes.

No quinto capítulo é descrita a dinâmica da proposta e as implementações necessárias para concretiza-la.

Por fim, no sexto capítulo encerra-se o trabalho com uma breve revisão das principais considerações, apresenta uma discussão dos resultados obtidos com a pesquisa e aponta trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo são abordados conceitos importantes para o entendimento das técnicas que são utilizadas neste trabalho. A Seção 2.1 é introduzido os algoritmos evolutivos, apresentando seus pontos fortes e suas influências para a otimização de problemas complexos. A seguir, na Seção 2.2 é introduzido os algoritmos de inteligência de enxame e seus pontos positivos na otimização de problemas dinâmicos. Por fim, na Seção 2.3 é apresentado a relação de intensificação e diversificação na convergência de algoritmos durante o processo de otimização.

2.1 Algoritmos Evolutivos

A natureza é uma fonte de inspiração para o desenvolvimento de vários algoritmo, como por exemplo os algoritmos evolutivos. A Computação Evolutiva se inspira no processo da seleção natural e evolução natural. Em termos gerais,um algoritmo evolucionário possui alguns componentes básicos para resolução de problemas (PARPINELLI; LOPES, 2011)

1. Várias representações para a possíveis soluções do problema;
2. Um modo de criar uma população inicial (aleatório ou determinístico);
3. Uma função que avalia a qualidade das soluções, ou seja, a aptidão do indivíduo (*Fitness*);
4. Um mecanismo de seleção para cruzamento;
5. Operadores evolutivos para criação de novas gerações (como mutação e *crossover*);
6. Parâmetros para controle do comportamento do algoritmo, como número de dimensões, controle de operadores e etc.

Nas subsecções estão apresentados os algoritmos do estado da arte que serão utilizados neste trabalho.

2.1.1 Algoritmo Genético

O Algoritmo Genético (*Genetic Algorithm* - GA) foi um dos primeiro algoritmos bioinspirados a serem propostos durante a década de 60 e 70, ele foi desenvolvido por Holland e seus colaboradores que tem como base a teoria da evolução de Darwin (BOOKER; GOLDBERG; HOLLAND, 1989) e é um dos algoritmos mais utilizados na Computação Evolutiva. A seleção natural de Darwin diz que o melhor indivíduo em uma determinada população, tem maiores chances de sobreviver e assim passar sua carga genética adiante, tornado assim a espécie mais apta às condições do ambiente. O GA utiliza essa seleção do indivíduo mais adaptado para direcionar a busca em direção das soluções ótimas.

A otimização do GA começa com a inicialização da população inicial, sendo de forma aleatória ou determinística, em que cada indivíduo possuí um cromossomo, que por sua vez representa uma possível solução para o problema. A partir dessa população inicia-se o primeiro ciclo evolutivo, em que cada indivíduo é avaliado, gerando assim um valor de aptidão (*fitness*) para ser usado na seleção da população. A seleção determina quais indivíduos irão cruzar gerando uma população intermediária, de modo que os mais adaptados tenham uma maior chance de serem selecionados. Os operadores evolutivos de cruzamento e mutação, são aplicados na população intermediária, em que o cruzamento tem o papel de intensificar a busca e a mutação o de diversificar a população. A partir deste ciclo o algoritmo se repete até que o critério de parada seja satisfeito, aplicando cada ciclo na população gerada pelo ciclo anterior.

2.1.2 Evolução Diferencial

A evolução diferencial (*Diferencial Evolution* - DE) e uma meta-heurística evolucionária para otimização de funções contínuas, proposta por Storn e Price em 1995 (PRICE; STORN; LAMPINEN, 2006). Seu nome se dá pela sua rotina de mutação que acontece por uma operação de subtração (diferenciação). O DE é amplamente estudado e as características que o tornam interessante são: Sua simplicidade, pois é um algoritmo com poucos operadores evolutivos e simples de serem implementados; Seu bom desempenho, tendo uma convergência rápida o DE se destaca entre vários outros algoritmos evolutivos; e pelo fato de possuir poucos parâmetros, tornando assim sua aplicação mais fácil e intuitiva.

O DE possui, basicamente, uma etapa de inicialização e três etapas em seu ciclo de evolução: mutação, cruzamento e seleção (nessa ordem). O processo evolutivo é iniciado após a

inicialização do algoritmo e é finalizado quando um determinado critério de parada for atingido. Primeiramente, o usuário define o tamanho da população (λ), taxa de chance de *crossover* (C_r) e fator de escala (F). Existem três tipos de vetores no DE que são: Vetor alvo, que é o vetor pai da geração atual; O vetor doador, que é gerado à partir da mutação; e o vetor *trial*, que é a combinação do vetor alvo com o vetor doador gerado pelo cruzamento.

Para criar o vetor doador, obtém-se uma amostra de três indivíduos distintos na população, assim é criado uma mutação à partir da diferença de dois destes vetores, multiplicada pelo fator de escala F e somada ao terceiro vetor. A etapa de mutação, responsável pela busca global do algoritmo (diversificação), e é o principal fator caracterizante do DE. O C_r controla o cruzamento que é responsável pela intensificação da busca e acontece à partir da recombinação dos vetores alvo e doador. Ao final a seleção acontece verificando uma melhora no *fitness* do vetor *trial* em relação ao alvo, caso não aconteça uma melhora a nova solução é descartada.

2.2 Algoritmos de Inteligência de Enxame

Os algoritmos de inteligência de enxame

2.2.1 Otimização por Enxame de Partículas

A otimização por enxame de partículas (*Particle Swarm Optimization* - PSO) foi proposta por Kennedy (EBERHART; KENNEDY et al., 1995). O PSO tem como inspiração o comportamento coordenado dos movimentos dos pássaros e cardumes de peixes. Cada partícula é uma solução potencial para o problema, representada por sua velocidade, localização no espaço de busca e uma memória que armazena a sua melhor posição visitada. O movimento de cada partícula depende de sua própria velocidade e da localização das boas soluções encontradas. O equilibro entre a intensificação e a diversificação é alcançada no PSO através do peso da inércia.

2.2.2 Otimização por Colônia de Bactérias

O algoritmo de Otimização por Colônia de Bactérias (*Bacterial Foraging Optimization* - BFO) foi proposto por Passino (PASSINO, 2002), inspirada no comportamento social de busca por alimento das bactérias Escherichia Coli. Durante o processo de busca a bactéria se move em pequenos passos enquanto procura alimento. Seu movimento é reali-

zado através de um conjunto de filamentos conhecido como flagelos, que ajudam a bactéria se mover em períodos alternados de natação (*swim*) e tombos (*tumble*). A alternância entre estes dois períodos chama-se quimiotaxia. Cada bactéria representa uma solução para o problema. O ambiente fornece o substrato para as bactérias interagirem e é representado pelo espaço de busca sendo otimizado. Quanto melhor for a região do espaço de busca, melhor será o resultado da função objetivo, e consequentemente, melhor será o substrato para as bactérias.

O BFO é composto por três rotinas principais: quimiotaxia, reprodução e eliminação-dispersão. Na quimiotaxia, uma bactéria com direção aleatória representa um tombo e uma bactéria com a mesma direção do passo anterior indicando uma execução. Na reprodução, a saúde de cada bactéria representa seu valor de *fitness*. Todas as bactérias são classificadas de acordo com seu estado de saúde e apenas a primeira metade da população sobrevive. As bactérias sobreviventes são divididos em dois filhos idênticos, de modo a formar uma nova população. O processo de eliminação-dispersão é responsável por aumentar a diversidade da população. A dispersão ocorre depois de um certo número de etapas de reprodução, quando algumas bactérias são escolhidos de acordo com uma probabilidade predefinida. Tais bactérias são mortas e os novos são gerados aleatoriamente em outra posição dentro do espaço de busca. A intecificação busca é realizado por ambos os passos de quimiotaxia e de reprodução.

2.2.3 Otimização por Colônia Vaga-lumes

O algoritmo de Otimização por Colônia de Vaga-Lumes (*firefly algorithm* - FA) por Yang (YANG, 2008). O FA trabalha com 3 regras principais para a otimização de funções *benchmarks*, sendo elas:

- Um vaga-lume será atraído pelo outro não importando o sexo deles.
- A atratividade entre dois vaga-lumes aumenta em relação a intensidade do brilho e decresce em relação ao aumento da distância entre eles.
- A proximidade do vaga-lume em relação a uma solução do espaço de busca influência na intensidade do brilho do mesmo.

Cada agente brilha proporcionalmente à qualidade da sua solução que, juntamente com a sua atratividade (β), ditam o quão forte ela atrai outros membros do enxame. Dois outros parâmetros definidos pelo usuário são o valor máximo de atração (β_i) e do coeficiente

de absorção (Υ), que determina a variação de atratividade com o aumento da distância da comunicação dos vaga-lumes. A variável de intensidade da luz β_i de cada vaga-lume, mantém o balanço entre a intensificação e diversificação da busca.

2.2.4 Otimização por Colônia de Morcegos

O algoritmo de Otimização por Colônia de Morcegos foi apresenta pela primeira vez por Yang (YANG; HOSSEIN GANDOMI, 2012). Aplicado aos mesmos *benchmarks* pode-se notar uma desempenho melhor do que os encontrados em AGs e ao PSO.

A ideia do algoritmo de morcegos é uma população de morcegos (possíveis soluções), em que é usado eco-localização dos morcegos, utilizado durante o seu voo para detectar presas e evitar obstáculos e uma rotina de voo aleatório para movimentação no espaço de busca, atualizando suas posições e suas velocidades. outros parâmetros são: o fator de decaimento de sonoridade (α) que atua em um papel semelhante a temperatura do *simulated annealing*, e o fator de aumento de pulso (γ) que regula a frequência de pulso. A atualização da taxa de pulso (R_i) e sonoridade (A_i) equilibra o comportamento intensificação e diversificação de cada morcego, respectivamente. De acordo com a diminuição da sonoridade ao morcego encontrar uma presa (solução ótima), a taxa de pulso vai aumentando para melhorar a acurácia do ataque.

2.2.5 Algoritmo de Busca por Cardume de Peixes

O algoritmo de Busca por Cardume de Peixes (*Fish School Search* - FSS) foi proposto por Carmelo (CARMELO FILHO, 2008). O processo de pesquisa do FSS é feito por uma população de indivíduos, com memória limitada, representado o peixe. O ambiente de busca é representado pelo mar em que os peixes se encontram, onde eles podem se movimentar para tentar melhorar sua solução. A densidade de comida representa a qualidade da solução, de forma que em um problema de maximização a densidade de comida é diretamente proporcional a qualidade do resultado. A densidade de comida de um local influencia o peixe através do peso, de forma que uma melhora continua da solução apresenta um peixe mais pesado.

Operadores Evolutivos

A versão do algoritmo estuda possui 4 operadores com suas funções bem definidas, que podem ser separados em duas classes (C JA FILHO, 2009): Alimentação (OA) e Movimentação. Para a movimentação temos as seguintes classificações: Operador de Movimento Individual (OMI); Operador de Movimentação Coletiva Instintiva (OMCI); e Operador de Movimentação Coletiva Volátil (OMCV).

Operador de Movimento Individual

Este OMI representa o movimento de cada um dos peixes individualmente, sem a influência do cardume. A próxima posição (candidata) é determinada pela geração de um número aleatório, de uma distribuição normal, e esse número é atribuído a uma das posições vizinhas do indivíduo. Após Isso o número é multiplicado por uma variável chama de $Step_ind$, que é a percentagem da amplitude do espaço de busca, essa variável decresce linearmente durante cada iteração, como mostrado na Equação 2.1.

$$\begin{aligned} &n_i(t) = x_i(t) + N(-1,1).Step_ind \\ &\Delta f = f(\vec{n}) - f(\vec{x}) \\ &\Delta x = \vec{n} - \vec{x} \\ &Step_ind(t+1) = Step_ind(t) - \frac{Step_indInicial - Step_indFinal}{iteracoes}; \end{aligned} \tag{2.1}$$

Em que n_i é a nova posição candidata, x_i é a posição atual, Δf é a variação do *fitness* e o Δx a variação do movimento. A nova posição é efetivamente atualizada, somente se, ela for melhor que a posição anterior do indivíduo. Sendo assim, pode-se observar que esse operador tem influência na busca global feita por cada indivíduo.

Operador de Alimentação

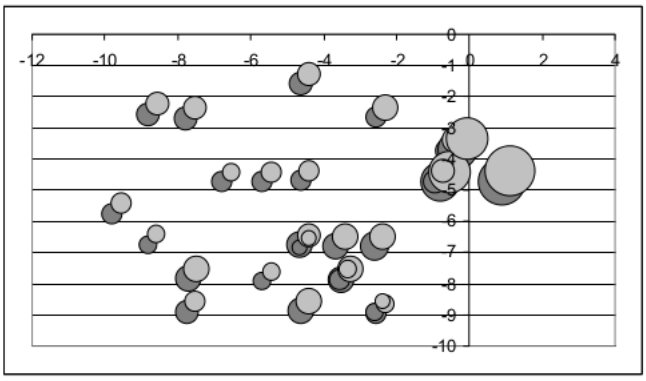
Quando são gerados, peixes possuem o mesmo peso, e ao longo da busca esse peso é alterado de acordo com o melhoramento da solução, ou seja, caso a solução melhore o peso aumenta. O peso de cada peixe representa o quão bem o indivíduo está na busca da solução. O cálculo do peso tem influência da população, utilizando maior variação de fitness encontrada entre todos os indivíduos como $\max \Delta f$, sendo a diferença entre o melhor e o pior indivíduos, mostrado na Equação 2.2.

$$W_i(t+1) = W_i(t) + \frac{\Delta t}{\max \Delta t} \tag{2.2}$$

Operador de Movimentação Coletiva Instintiva

Somente os indivíduos que realizaram o OMI, ou seja, Δx tem que ser diferente de zero, terão influência no cálculo do movimento coletivo. Após o calculo todos os indivíduos são atualizados, mesmo os que não realizaram o movimento individual. Esse operador coletivo instintivo tem como contribuição a busca local fazendo com que a população inteira se direcione para um espaço de busca possivelmente melhor, simulando assim, o efeito de um cardume de peixes como pode ser visto da Figura 2.1.

Figura 2.1: Representação gráfica da influência do Movimento Coletivo Instintivo na população

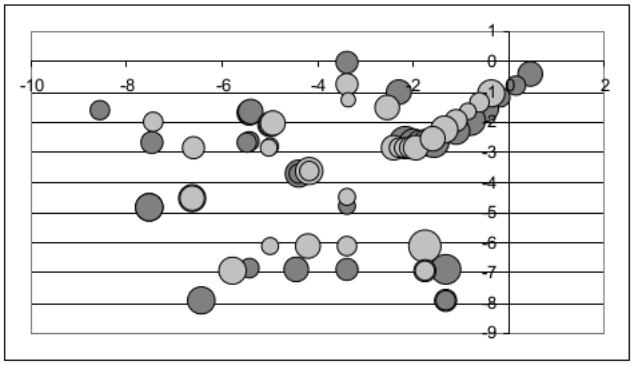


Fonte: C JA Filho (2009).

Operador de Movimentação Volátil Coletiva

Este movimento é baseado na melhora em geral da população, que é representada através do peso médio da população. Quando tem-se uma melhora nas soluções ocorre a contração da população em relação ao baricentro dos indivíduos, caso contrário, ocorre a dilatação. O cálculo deste baricentro é feito com base na posição de cada indivíduo e o peso de cada um deles. Todos os indivíduos da população são afetados pela contração ou dilatação, de forma que a busca pode ficar mais focada em um ponto, ou expandida para buscar mais resultados, como mostrado na Figura 2.2. Esse operador tem como maior contribuição tanto a manutenção da variabilidade genética, de forma que ajuda a busca a não ficar presa em um ótimo local, quanto a intensificação da mesma, pois quando acontece uma contração a busca naquele local fica mais concentrada.

Figura 2.2: Representação gráfica da influência do Movimento Coletivo Instintivo de uma contração



Fonte: C JA Filho (2009).

2.3 Intensificação e Diversificação

Importância da diversificação e diversificação na otimização de problemas dinâmicos e técnicas utilizadas para mante-las.

entropy: (MORI; KITA; NISHIKAWA, 2001) hamming distance: (RAND; RIOLO, 2005) memory-of-inertia: (MORRISON; DE JONG, 2001) peak cover: (BRANKE, 2012) maximun-spread: (GOH; TAN, 2009)

3 Problemas Abordados

Nessa seção será apresentada uma breve introdução aos *benchmarks* que serão utilizados para análise nesse trabalho, junto a isso é feito uma introdução aos problemas dinâmicos, suas variações e dificuldades.

Durante o dia a dia, pode-se observar diversos problemas complexos em que as condições do ambiente podem mudar com o passar do tempo, sendo assim uma solução válida com o passar do tempo, acaba não sendo mais suficientemente boa para suprir as necessidades das novas condições (BRANKE, 2012).Para problemas com mudanças em seu ambiente é necessário um sistema de resolução que se adapte as alterações e esteja sempre apto a procurar por soluções melhores.

Na computação existem várias áreas que se dedicam a otimizar problemas desse tipo como por exemplo os EA e os IS. Para que poder avaliar os algoritmos utilizados na otimização destes problemas são geradas funções *benchmarks* que apresentam características dos problemas reais, como por exemplo mudança de elevação de terrenos.

3.1 Funções *Benchmark*

O uso das funções *benchmark* é um ponto crucial para o desenvolvimento, comparação, avaliação e otimização dessa classe de problemas (NGUYEN; YANG; BRANKE, 2012). O fato das condições do problema variarem de acordo com o tempo acrescentam uma camada de dificuldade a mais para obter uma solução satisfatória após uma mudança. Um bom *benchmark* possui as seguintes características.

1. Flexibilidade: Configurável sobre diferentes abordagens dinâmicas, como por exemplo, a frequência ou intensidade das alterações no ambiente, em diferentes escalas.

2. Simplicidade e Eficiência: Deve ser fácil de implementar e analisar.

3. Associação com Problemas Reais: Ter a capacidade de fazer previsões em problemas reais ou se assemelhar em alguma extensão.

Para a analisar um *benchmarks* é necessário dar atenção para o tipo de dinamismo que esse problema apresenta, pois para otimiza-lo são escolhidas estratégias de otimização baseadas no tipo de dinamismo. Cada *benchmarks* possui suas características e com isso cada algoritmo aplicado a ele terá que analisar como está abordando essas peculiaridades. A Características observadas nas funções encontradas são (CRUZ; GONZÁLEZ; PELTA, 2011):

1. Influência Temporal: Sempre que uma solução futura depende de outra encontrada anteriormente pelo algoritmo, ou seja, quando o tipo de mudança que o algoritmo vai sofrer depende de quais soluções foram encontradas ao decorrer da otimização. Neste caso os resultados podem mudar para cada tentativa de otimização.

2. Previsibilidade: Quando as mudanças geradas pelo problema seguem um padrão que pode ser previsto durante a otimização do problema. Podem ser alterações por mudança em uma escala fixa, incremental, cíclica, periódica.

3. Visibilidade: Sempre que as mudanças são visíveis pelo algoritmo que está otimizando o problema, de foram que o algoritmo não precise usar muitos detectores para perceber a mudança no ambiente. Podem ser pontos específicos do espaço, que são detectados ao re-avaliar a função objetivo, ou até mesmo, as restrições.

4. Problemas com Restrições: As mudanças do problema podem ocorrer nas restrições, que podem mudar ao longo do tempo (a variável de tempo influencia na função de restrição) ou o podem ser ativadas e/ou desativadas durante a otimização.

5. Número de Objetivos: São as problemas multi-objetivos, que neste caso podem alterar o número de objetivos durante a otimização, chegando até, ao ponto de tem somente um objetivo em vista.

6. Intensidade das Mudanças: qual a escala da mudança, ou qual o período de ciclo das mudanças, ou o limite das mudanças.

7. Fator da Mudança: Onde o dinâmismo pode ocorrer, sendo: Função objetivo; Domínio das variáveis; Número de variáveis; Restrições; ou outros parâmetros.

3.1.1 Avaliação de Desempenho

Para avaliar um *benchmarks* são estipuladas métricas que serão utilizadas para medir a qualidade dos resultados obtidos pelo algoritmo de otimização. Os métodos de avaliação

apresentados foram selecionados por serem amplamente utilizados em algoritmos bioinspirados aplicados em ambientes dinâmicos e possivelmente serão usados para avaliar o desempenho dos algoritmos selecionados neste trabalho. A forma de avaliação pode ser dividida em dois grandes grupos: Medida de desempenho baseados em otimalidade e os em comportamento.

Otimalidade

As medidas de desempenho baseadas na otimalidade avaliam a capacidade dos algoritmos em encontrar as soluções com o melhor valor de *fitness* ou as que estão mais próximas do ótimo global (medidas baseadas na distância) e essa avaliação leva em conta o algoritmo como um todo. Entre elas tem-se:

- Melhor da Geração: Avalia qual o melhor indivíduo de cada iteração do algoritmo, gerando assim uma curva de desempenho que apresenta os pontos de melhora do algoritmo e o quão perto está do ótimo do problema. Esse é o método mais frequentemente utilizado.
- Melhor Erro Antes da Mudança: Calcula a média do erro (diferença da melhor solução atual para o ótimo do problema) da população logo antes da mudança do ambiente. Este caso só pode ser usado quando o dinamismo do *benchmark* gerado é previsível ou facilmente identificável, que por sua vez, dependo de como o *benchmark* muda o ambiente.
- Pontuação Normalizada: Mede a diferença entre uma determinada gama de algoritmos em relação a uma lista pré-determinada de problemas, fazendo com que cada algoritmo execute cada problema um número determinado de vezes. Com os resultados normalizados, para a melhor solução é atribuído o valor 1(um) e para a pior solução é atribuído o valor 0(zero).

Comportamento

As medidas de desempenho baseadas em comportamento avaliam a influência de certos comportamentos que são importantes na hora de manter um nível de *fitness*, como por exemplo:

- Manutenção da Diversidade: como o nome já diz, mede quanto que um comportamento pode ajudar na manutenção da diversidade da população ao longo do processo de otimização. Este é um fator muito estudado em ambientes dinâmicos, pois com

uma grande diversidade da população existem mais chances de se encontrarem os novos ótimos globais.

- Velocidade de Convergência Após uma Mudança: mede quão rápido o sistema se adapta após perceber uma alteração. Essa medida de comportamento não se aplica a sistemas que tem dificuldades em perceber uma alteração no ambiente. Para analisar essa medida é feita uma análise de quantas iterações são necessárias para se encontrar um novo ótimo, em relação a distância.

- Performance Após uma Mudança: mede a capacidade do algoritmo de buscar novos resultados após as mudanças no ambiente, de forma que restringe a queda do *fitness* quando percebe uma piora. Essa medida também é conhecida como medida de estabilidade ou satisfabilidade.

3.2 Problemas Dinâmicos com domínio contínuo

Nesta etapa serão apresentados e detalhados os problemas que serão abordados no trabalho. Os problemas selecionados para esse trabalho serão usados para realizar uma análise comparativa com outras abordagens encontradas na literatura e também testar o desempenho de cada operador evolutivo que os algoritmos irão abordar. A maioria dos problemas selecionados não possuem um *link* temporal com as soluções anteriores. São apresentados com e sem restrições na função objetivo ou no limite do domínio das variáveis, tendo em sua maioria dinamismos previsíveis, periódicos e constantes.

3.2.1 *Moving Peaks*

O *benchmark* de movimentação de picos (*moving peaks*) foi desenvolvido por Jürgen Branke (BRANKE, 1999) e tem como principal ideia gerar um problema que possa ser amplamente utilizado em algoritmos bioinspirados, pois a maioria dos problemas dinâmicos, até o dado momento, sofriam uma alteração que gerava um novo ambiente totalmente diferente do anterior, tornando assim um recomeço da otimização uma solução melhor.

A ideia é gerar um *benchmark* que, ao sofrer uma mudança no ambiente, ficasse levemente alterado, o suficiente para que as soluções anteriores ajudassem a encontrar as novas geradas. Para isso foi criado o *moving peaks* com a ideia de ter uma paisagem multidimensional

artificial que consiste em vários picos, em que a altura, a largura e a posição de cada pico é ligeiramente alterada cada vez que uma mudança no ambiente ocorre.

A função objetivo é dada pela Equação 3.1, na qual tem-se H como altura, W como peso, n como o número de picos e m como o número de dimensões. Todos esses parâmetros são definidos previamente ao testes.

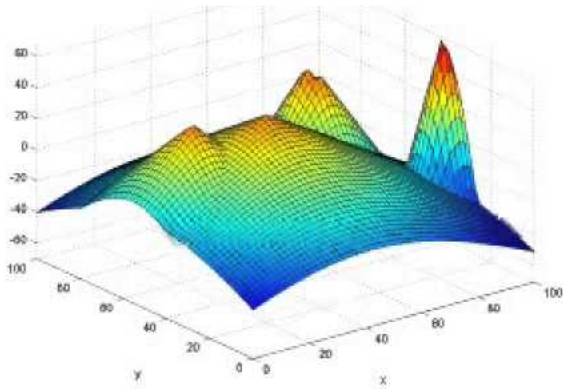
$$F(\vec{x}, t) = \max_{i=1 \to n} \frac{H_i(t)}{1 + W_i(t) \sum_{j=1}^{m} (x_j - X_j(t))^2} \tag{3.1}$$

A cada Δe (valor que representa o intervalo/frequência das mudanças do ambiente), a altura e o peso são alterador adicionando uma variável Guassiana aleatória. O local de cada um dos picos é movido por um vetor $\vec{v}$ a uma distância s fixa para uma direção aleatória. Essas alterações podem ser descritas pela Equação 3.2.

$$\begin{aligned} &\sigma \in N(0,1) \\ &H_i(t) = H_i(t-1) + 7.\sigma \\ &W_i(t) = W_i(t-1) + 0,01.\sigma \\ &\vec{X}_i(t) = \vec{X}_i(t-1) + \vec{v} \end{aligned} \tag{3.2}$$

Um exemplo de como as máximas se movem ao longo do tempo em um espaço bidimensional pode ser visto na Figura 3.1

Figura 3.1: Gráfico de duas dimensões do movimento dos picos, após 300 alterações no ambiente, $s = 0,9$



Fonte: Branke (1999).

3.2.2 *Ocillating Peaks*

O *benchmark* de oscilação de picos (*ocillating peaks*) é baseado no de movimentação de picos, porem foi desenvolvido para algoritmos evolutivos com utilização de memória.

É uma combinação linear entre duas funções em que o peso delas se alteram ao longo do tempo, portanto, a função G, começa em g_1, passa para g_2, depois volta para g_1 e assim sucessivamente. Em outras palavras, o máximo absoluto de G oscila entre dois pontos g_1 e g_2. As funções que representam as alteração da função G estão representadas na Equação 3.3, em que, $\lambda(t)$ representa a oscilação da função, n é a soma das picos das duas funções que compõem a combinação linear e m é o número de dimensões.

$$
\begin{aligned}
&\delta \in N(0,1) \\
&\lambda(t) = \frac{\cos\frac{2\pi t}{100\delta} + 1}{2} \\
&g_1(\vec{x}) = \sum_{i=1}^{n/2} \frac{H_i}{1 + W_i \sum_{j=1}^{m}(x_j - X_j(t))^2} \\
&g_2(\vec{x}) = \sum_{i=n/2+1}^{n} \frac{H_i}{1 + W_i \sum_{j=1)}^{m}(x_j - X_j(t))^2} \\
&G(t,\vec{x}) = \lambda(t) g_1(\vec{x}) + (1 - \lambda(t)) g_2(\vec{x})
\end{aligned}
\tag{3.3}
$$

4 Trabalhos Relacionados

A Computação Natural possuí vários operadores diferentes em algoritmos diferentes para serem aplicados em diversos tipos de problemas. Nesta Seção são apresentados os trabalhos do estado da arte encontrados na literatura que utilizam esses algoritmos bioinspirados em problemas dinâmicos de domínio contínuo.

Os trabalhos apresentados nesta seção são utilizado para analisar a aplicação destes algoritmos em diferentes ambiente e com diferentes condições, conseguindo assim extrair informações de influência de cada operador evolutivo utilizado na otimização dos problemas e/ou re-otimização.

4.1 Comportamento do AG em ambientes dinâmicos

No trabalho de (RAND; RIOLO, 2005) é feito uma análise do comportamento do AG em ambientes dinâmicos e mostra que na maioria dos trabalhos que analisam o AG, somente a performance do algoritmo é analisada, ou seja, o quão perto do melhor resultado chegou. Então são analisados quatro fatores principais para determinar a eficiência do AG em ambientes dinâmicos neste trabalho, sendo eles:

1. Performance: Para entender a performance do algoritmo existe duas vertentes, sendo uma a avaliação do melhor indivíduo da população para cada iteração do AG, e a outra é avaliar a média da população em para cada uma das iterações. Para a aplicar AG no SL-HDF é usado a melhor solução antes da primeira alteração como média inicial.

2. Satisfabilidade: É a medida da habilidade do sistema de manter um certo nível de *fitness* no decorrer da otimização e não deixar esse nível cair abaixo de um determinado limite. Esta medida não necessariamente representa o quão rápido (menos interações necessárias) o sistema chega em uma nova ótima solução, e sim se ele consegue manter um nível de *fitness* da população.

3. Robustez: É a medida de como o sistema reage a uma alteração, de forma que ao sofrer uma alteração o *fitness* não pode ter uma queda muito brusca. A medida de robustez usada

neste trabalho foi a média do *fitness* no estado atual do ambiente sobre a média do *fitness* no estado anterior do sistema, para uma alteração perceptível.

4. Diversidade: É a medida que representa a variação do genoma da população, de forma que uma população que possuí uma alta diversidade tem maiores chances de encontrar novas solução e assim se adaptar melhor a uma mudança do ambiente. Existem várias técnicas estudadas para manter a diversidade da população durante o processo evolutivo, e para medir a diferença genotípica é usado a distância de *Hamming*.

Na aplicação do AG no SL-HDF pode-se notar que a performance do algoritmo no ambiente dinâmico é superior sua aplicação em ambientes estáticos quando há um grande número de iterações. Inicialmente o ambiente dinâmico perde para o estático mas a partir da metade do processo evolutivo o dinâmico gera um *fitness* maior no melhor indivíduo e na média da população.

A análise da satisfabilidade mostra que em relação a aplicação em um ambiente estático o ambiente dinâmico tem quase o mesmo nível médio de *fitness* porém o ambiente dinâmico está sendo recompensado por blocos de construção intermediária diferentes e, portanto, tem uma pressão seletiva superior para encontrá-los.

Na analise de robustez pode-se notar que a cada 100 iteração (quando ocorre uma mudança no ambiente) exite uma queda na média do *fitness*, porém o sistema se recupera rapidamente, e a cada nova mudança e queda do *fitness* diminui.

A diversidade do sistema teve um comportamento inesperado, pois inicialmente achava-se que o sistema iria perder a diversidade até uma mudança ocorrer e depois a diversidade iria aumentar, porém aconteceu exatamente o contrário, tendo que após uma mudança a diversidade diminui e vai aumentando até identificar uma nova mudança.

4.2 Comportamento do PSO em ambientes dinâmicos

O PSO(CARLISLE, 2002)

(SHI; EBERHART, 1998)

5 Proposta

Proposta

6 Considerações

Considerações

Bibliografia

ANDRÉ, L.; STUBS PARPINELLI, R. The multiple knapsack problem approached by a binary differential evolution algorithm with adaptive parameters. *Polibits*, Instituto Politécnico Nacional, Centro de Innovación y Desarrollo Tecnológico en Cómputo, n. 51, p. 47–54, 2015.

BELLMAN, R. E.; DREYFUS, S. E. *Applied dynamic programming*. : Princeton university press, 2015. 200–206 p.

BOOKER, L. B.; GOLDBERG, D. E.; HOLLAND, J. H. Classifier systems and genetic algorithms. *Artificial intelligence*, Elsevier, v. 40, n. 1, p. 235–282, 1989.

BOUSSAÏD, I.; LEPAGNOT, J.; SIARRY, P. A survey on optimization metaheuristics. *Information Sciences*, Elsevier, v. 237, p. 82–117, 2013.

BRANKE, J. Memory enhanced evolutionary algorithms for changing optimization problems. In: CITESEER. *In Congress on Evolutionary Computation CEC99*. 1999.

BRANKE, J. *Evolutionary optimization in dynamic environments*. : Springer Science & Business Media, 2012.

C JA FILHO, B. et al. On the influence of the swimming operators in the fish school search algorithm. In: IEEE. *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. 2009. p. 5012–5017.

CARLISLE, A. J. *Applying the particle swarm optimizer to non-stationary environments*. : Auburn University, 2002.

CARMELO FILHO, J. et al. A novel search algorithm based on fish school behavior. In: IEEE. *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*. 2008. p. 2646–2651.

CASTRO, L. N. de. Fundamentals of natural computing: an overview. *Physics of Life Reviews*, Elsevier, v. 4, n. 1, p. 1–36, 2007.

CAVALCANTI-JÚNIOR, G. M. et al. A hybrid algorithm based on fish school search and particle swarm optimization for dynamic problems. In: *Advances in Swarm Intelligence*. : Springer, 2011. p. 543–552.

CRUZ, C.; GONZÁLEZ, J. R.; PELTA, D. A. Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Computing*, Springer, v. 15, n. 7, p. 1427–1448, 2011.

EBERHART, R. C.; KENNEDY, J. et al. A new optimizer using particle swarm theory. In: NEW YORK, NY. *Proceedings of the sixth international symposium on micro machine and human science*. 1995. v. 1, p. 39–43.

GOH, C.-K.; TAN, K. C. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, IEEE, v. 13, n. 1, p. 103–127, 2009.

HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* : U Michigan Press, 1975.

LI, C.; YANG, M.; KANG, L. A new approach to solving dynamic traveling salesman problems. In: *Simulated Evolution and Learning*. : Springer, 2006. p. 236–243.

MADEIRO, S. S. et al. Density as the segregation mechanism in fish school search for multimodal optimization problems. In: *Advances in Swarm Intelligence*. : Springer, 2011. p. 563–572.

MORI, N.; KITA, H.; NISHIKAWA, Y. Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm. *Transactions of the Institute of Systems, Control and Information Engineers*, v. 14, p. 33–41, 2001.

MORRISON, R. W. Performance measurement in dynamic environments. In: CITESEER. *GECCO workshop on evolutionary algorithms for dynamic optimization problems*. 2003. p. 5–8.

MORRISON, R. W.; DE JONG, K. A. Measurement of population diversity. In: SPRINGER. *Artificial Evolution*. 2001. p. 31–41.

MOSER, C. I. Review all currently known publications on approaches which solve the moving peaks problem. Citeseer, 2007.

NGUYEN, T. T.; YANG, S.; BRANKE, J. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, Elsevier, v. 6, p. 1–24, 2012.

OLIVEIRA, M. C. S. de; SILVA, T. L.; ALOISE, D. J. Otimização por nuvem de partículas: diferença entre aplicações a problemas contínuos e discretos. *XXXVI SBPO. São João Del-Rei–MG*, 2004.

PARPINELLI, R. S.; LOPES, H. S. New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation*, Inderscience Publishers Ltd, v. 3, n. 1, p. 1–16, 2011.

PASSINO, K. M. Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems, IEEE*, IEEE, v. 22, n. 3, p. 52–67, 2002.

PRICE, K.; STORN, R. M.; LAMPINEN, J. A. *Differential evolution: a practical approach to global optimization*. : Springer Science & Business Media, 2006.

RAND, W.; RIOLO, R. Measurements for understanding the behavior of the genetic algorithm in dynamic environments: A case study using the shaky ladder hyperplane-defined functions. In: ACM. *Proceedings of the 7th annual workshop on Genetic and evolutionary computation*. 2005. p. 32–38.

ROZENBERG, G.; BCK, T.; KOK, J. N. *Handbook of natural computing*. : Springer Publishing Company, Incorporated, 2011.

SHI, Y.; EBERHART, R. A modified particle swarm optimizer. In: IEEE. *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. 1998. p. 69–73.

YANG, X.-S. Firefly algorithm. *Nature-inspired metaheuristic algorithms*, v. 20, p. 79–90, 2008.

YANG, X.-S.; HOSSEIN GANDOMI, A. Bat algorithm: a novel approach for global engineering optimization. *Engineering Computations*, Emerald Group Publishing Limited, v. 29, n. 5, p. 464–483, 2012.